תרגיל בית 1: נער שליחויות

מטרות התרגיל

- נתמודד עם בעיות פרקטיות ותיאורטיות של חיפוש במרחבי מצבים עצומים.
- נתרגל את הנלמד בהרצאות ובתרגולים.
- נתנסה בתכנות ב- python לפתרון בעיות פרקטיות.

הנחיות כלליות

- **תאריך הגשה:** יום ראשון, 02.12.2018, בשעה 23:59.
- את המטלה יש להגיש **בזוגות בלבד**.
- יש להגיש מטלות מוקלדות בלבד. פתרונות בכתב יד לא ייבדקו.
- ניתן לשלוח שאלות בנוגע לתרגיל לתיבת המייל הקורסית: ai.technion@gmail.com.
- בקשות דחיה **מוצדקות** יש לשלוח למיכל בדיאן בלבד.
- במהלך התרגיל ייתכן שנעלה עדכונים, תיקונים והבהרות לדף FAQ ייעודי באתר. העדכונים הינם **מחייבים**, ועליכם להתעדכן דרך עמוד זה.
- שימו לב, התרגיל מהווה כ- 13% מהציון הסופי במקצוע ולכן העתקות תטופלנה בחומרה.
- ציון המטלה יורכב מהגורמים הבאים:
  - **הדו"ח הסופי.** מעבר לתשובות הנכונות, אתם נבחנים גם על הצגת הנתונים והתוצאות בצורה קריאה ומסודרת. בפרט, יש לכתוב כותרות מתאימות לגרפים ולצירים בגרפים (כולל יחידות מידה היכן שצריך).
  - **הקוד המוגש.** יש להקפיד על הגשת קוד מסודרת בהתאם להנחיות. יש לכתוב הערות במקומות חשובים בקוד כדי שיהיה קריא וקל לבדיקה.
- גרסת python איתה אתם נדרשים לעבוד הינה 3.6. גם קבצי המקור שקיבלתם מתאימים לגרסה זו.
- אלא אם נכתב אחרת, אין לשנות פונקציות מוכנות שקיבלתם, או את החתימה של פונקציות שהתבקשתם לממש. בפרט, אין לשנות תוכן קבצים אותם אינכם מתבקשים להגיש. בנוסף, אין ליצור קבצים חדשים, אלא לערוך את הקבצים שהתבקשתם במפורש בלבד. אם יש בעיה נקודתית, ניתן לשלוח מייל לתיבה הקורסית.
- לצורך ההרצות תצטרכו להתקין את החבילות הבאות של python: numpy, scipy, matplotlib. חבילות אלו מותקנות כברירת מחדל עם ההתקנה של Anaconda.
- בתרגילי הבית בקורס הרצת הניסויים עשויה לקחת זמן רב, ולכן מומלץ מאוד להימנע מדחיית העבודה על התרגיל לרגע האחרון. לא תינתנה דחיות על רקע זה.

הבהרות ועדכונים שנוספים אחרי הפרסום הראשוני יסומנו בצהוב.

פרק ראשון – משלוחי פיצה (90 נק')

חלק א' – מבוא והנחיות (3 נק')

במטלה זו נעסוק בהפעלת אלגוריתמי חיפוש על מרחבי מצבים גדולים במיוחד לבעיות ניווט. מומלץ לחזור על שקפי ההרצאות והתרגולים הרלוונטיים לפני תחילת העבודה על התרגיל.

במהלך התרגיל תתבקשו להריץ מספר ניסויים ולהריץ את תוצאותיהם. אתם נדרשים לבצע ניתוח מעמיק ומפורט של התוצאות, כפי שיוסבר בהמשך.

מוטיבציה

במקביל ללימודיו בטכניון, שלומי עובד כשליח פיצה. במהלך המשמרת שלו, שלומי מקבל מספר הזמנות אותן הוא צריך לפזר במקומות שונים במפה. בכל פעם הוא מקבל x הזמנות ויוצא לדרך במטרה להגיע לכל יעד ברשימה. כידוע, גובה הטיפ שיקבל תלוי בזמן שייקח לו להעביר את כל הפיצות ללקוחות ולכן הוא רוצה לעשות זאת בזמן הקצר ביותר. מאחר והפיצרייה מאוד פופולרית, יש הרבה מאוד משלוחים שעליו לבצע ומיכל הדלק בטוסטוס שלו לא תמיד מספיק בכדי להגיע לכולם מבלי לעצור ולתדלק בדרך. למזלו, חברים של שלומי (זה אתם!) במקרה לוקחים הסמסטר את הקורס "מבוא לבינה מלאכותית". שלומי מבקש מכם לעזור לו לתכנן מראש את הדרך היעילה ביותר להגיע לכל הלקוחות מבלי להיתקע בלי דלק בדרך, בהינתן רשימת יעדים, מיקומים של תחנות דלק במפה וגודל מיכל הדלק שלו.

פורמאליזם

נתונה רשת כבישים בצורת גרף $G_M = (V, E)$ שבו כל צומת (vertex) מייצג צומת דרכים (junction), והקשתות (edges) מייצגות דרך המקשרת בין צמתי דרכים (links). בנוסף, נתונה רשימה של צמתים על המפה בהן נמצאות תחנות דלק.

נתונה נקודת מוצא על הגרף $v_0 \in V$, וכן נתונות $k \in \mathbb{N}$ הזמנות: $Ord = \{t_1, \dots, t_k\}$, כאשר הזמנה i היא צומת ברשת הכבישים $t_i \in V$. בנוסף, נתונות $l \in \mathbb{N}$ תחנות דלק $GasStations = \{f_1, \dots, f_l\}$ שניתן לתדלק בהן, כאשר תחנת תדלוק j היא גם כן צומת ברשת הכבישים $f_j \in V$.

בתחילת הנסיעה, לטוסטוס יש מספיק דלק כדי שיספיק לעוד d_0 קילומטרים. לאחר נסיעה של x קילומטרים, וטרם מעבר בתחנת דלק כלשהי, יישאר במיכל מספיק דלק לעוד $d_0 - x$ קילומטרים. בכל פעם שאנו עוצרים בתחנת דלק, מיכל הדלק מתמלא והטוסטוס יכול לנסוע d_{refule} קילומטרים החל מנקודה זו (בלי קשר לכמה קילומטרים נשארו במיכל הדלק טרם התדלוק). נמדוד את יתרת הדלק במיכל ע"פ המרחק המקסימלי שניתן לנסוע בעזרתו.

לצורך פשטות, ניתן להניח שאין הזמנות ו/או תחנות דלק שממוקמות בנקודת המוצא v_0 וכן הצמתים המתאימים לתחנות הדלק זרים מהצמתים המתאימים להזמנות (כלומר $\forall_{i\in[k]}\forall_{j\in[l]}: t_i \neq f_j \wedge t_i \neq v_0 \wedge f_j \neq v_0$).

אנו רוצים להחזיר מסלול נסיעה חוקי P הממלא את כל דרישות הלקוחות:

1. P מתחיל בנקודת המוצא v_0.
2. P עובר בכל ההזמנות.
3. בכל רגע במסלול, לטוסטוס לא נגמר הדלק. כלומר, אסור שמספר הקילומטרים שנשארו במיכל יהיה שלילי. לצורך פשטות נניח שאם הגענו לתחנה האחרונה או לתחנת דלק עם בדיוק 0 ק"מ זה תקין.

את איכות המסלול P שיחושב ע"י התוכנית נמדוד לפי מרחק הנסיעה הכולל, כפי שיפורט בהמשך.

הבנת הבעיה

אנו מחפשים למעשה פרמוטציה מהצורה $v_0, v_1, \dots, v_r$ כאשר $Ord \subseteq \{v_1, \dots, v_r\} \subseteq Ord \cup GasStations$, שמקיימת את הדרישות הכתובות לעיל. עבור פרמוטציה כלשהי, נרצה למצוא את המסלול הקצר ביותר האפשרי בין כל זוג צמתים בעזרת אלגוריתם A*, שכפי שראינו בהרצאות, שהוא אלגוריתם לחיפוש מיודע. ניתן לראות שאפילו מבלי להתחשב באילוץ הדלק ישנן $k!$ פרמוטציות כאלה. כאשר נוסיף את אילוץ הדלק הבעיה מסתבכת אפילו יותר. אם נניח לצורך הדוגמה שידוע מראש שבין כל זוג הזמנות אנו צריכים לעבור בתחנת דלק אחת מתוך l תחנות אפשריות, נקבל שמספר הפרמוטציות הוא $(k!) \cdot l^{k-1}$. מכאן שעבור כמות הזמנות לא גדולה במיוחד, לא נוכל לפתור את הבעיה בעזרת חיפוש brute force.

במהלך המטלה ניגש לבעיה מזוויות שונות וננסה למצוא פתרונות עבורה בעזרת כלים שלמדנו בקורס.

תרגיל

1. הזינו טבלה של מספר הפרמוטציות האפשריות עבור ערכי k (מספר הזמנות) מ-1 עד 10 ללא אילוץ הדלק ובעמודה נוספת עם אילוץ הדלק תחת ההנחה שתיארנו לעיל, כאשר $l = 5$. היעזרו בנוסחאות דלעיל.

חלק ב' – הגדרת מרחב החיפוש במפה

בהינתן רשת הכבישים, נקודת מקור $u \in V$ ונקודת יעד $v \in V$, נגדיר מרחב חיפוש עבור מציאת מסלול ביניהן:

$$Map \triangleq (S_{map}, O_{map}, I_{map}, G_{map})$$

מרחב החיפוש שיווצר יהיה קצת טריוויאלי וכמעט זהה לגרף של רשת הכבישים, אך השמירה על כלליות תאפשר לנו לממש את האלגוריתמים באופן כללי יותר.

- **קבוצת המצבים:**
נרצה לייצג מצב כך שיחזיק את כל המידע שנחוץ לנו על נקודת זמן בפתרון, ושניתן להמשיך ממנו הלאה (המסלול הסופי יימצא על ידי מעבר על ההורים בעץ החיפוש). במקרה המדובר מספיק לשמור את הצומת בגרף בו אנו נמצאים.

$$S_{map} \triangleq \{(v) | v \in V\}$$

- **קבוצת האופרטורים:**
ניתן לעבור ממצב אחד למשנהו בתנאי שיש כביש מהצומת המייצג את המצב הראשון לצומת המייצג את המצב השני.

$$O_{map} \triangleq \{(S_1, S_2) | (S_1.v, S_2.v) \in E\}$$

  - **עלות אופרטור:**
במטלה נגדיר המחיר של מעבר מצומת דרכים אחד לצומת דרכים אחר ע"י האורך של הכביש ביניהם.

$$cost((S_1, S_2)) = length((S_1.v, S_2.v))$$

- **המצב ההתחלתי:**

$$I_{map} \triangleq (u)$$

- **מצבי המטרה:**

$$G_{map} \triangleq \{(v)\}$$

חלק ג' – הגדרת מרחב החיפוש של מסלולי נסיעת הטוסטוס (15 נקודות)

בהינתן רשת הכבישים, נקודת המוצא ורשימת ההזמנות, נגדיר מרחב חיפוש עבור בעיית המשלוחים:

$$Deliveries = (S_d, O_d, I_d, G_d)$$

- **קבוצת המצבים:**

נרצה לייצג מצב כך שיחזיק את כל המידע שנחוץ לנו על נקודת זמן בפתרון, ושניתן להמשיך ממנה הלאה (אך את הפתרון נמצא למשל על ידי מעבר על ההורים בעץ החיפוש).

$$S_d \triangleq \left\{ \left(\underbrace{v}_{\substack{\text{המיקום}\\ \text{הנוכחי}}}, \underbrace{d}_{\substack{\text{כמה ק"מ}\\ \text{דלק נשארו}}}, \underbrace{T}_{\substack{\text{קבוצת הזמנות}\\ \text{מחכות}}}, \underbrace{F}_{\substack{\text{קבוצת הזמנות}\\ \text{גמורות}}} \right) \middle| \begin{array}{c} v \in Ord \cup GasStations \\ T, F \subseteq [k] \\ d \geq 0 \\ T \cap F = \emptyset \\ T \cup F = [k] \end{array} \right\}$$

הערה: מספיק לשמור במצב רק קבוצה אחת מהשתיים (T, F), ואת השנייה להסיק מהקבוצה השמורה ומקבוצת ההזמנות ההתחלתית Ord. לצורך נוחות הדיון נשמור את שתי הקבוצות בכל מצב.

- **קבוצת האופרטורים:**

<u>אופרטורים עבור הורדת הזמנה:</u>

ניתן לעבור ממצב אחד למשנהו בתנאי שהזמנה t_i מחכה ונמצאת בקבוצה T (של המצב עליו פועל האופרטור), וכן בתנאי שמיכל הדלק של מצב המקור הכיל מספיק מרחק נסיעה בכדי להגיע לצומת t_i מהצומת שמיוצג ע"י מצב המקור.

<u>סימון</u>: מדד למרחק בין זוג צמתים $v_1, v_2 \in V$ יסומן ב- $Dist(v_1, v_2)$. זוהי למעשה פונק' $Dist: V \times V \mapsto \mathbb{R}$.

<u>הערה</u>: שימו לב שניתן להעלות על הדעת לפחות 2 הגדרות שונות עבור הסימון $Dist$: (1) אורך המסלול הקצר ביותר מ- v_1 ל- v_2 על גבי מפת הכבישים; (2) המרחק האווירי בין הצמתים v_1, v_2. לצורך ההגדרות הבאות נשתמש בסימון $Dist$ ללא הגדרה מפורשת של פונק' כזו. בהמשך נציג שתי הגדרות שונות עבור $Dist$, שכל אחת מהן תגדיר בעיה תואמת.

לכל $i \in [k]$ נגדיר את האופרטור o_{t_i} באופן הבא:

$$\forall_{(u,d,T,F) \in S}: o_{t_i}((u, d, T, F)) \triangleq \begin{cases} (t_i, d - Dist(u, t_i), T \setminus \{i\}, F \cup \{i\}) & ; \quad i \in T \wedge d \geq Dist(u, t_i) \\ \emptyset & ; \quad otherwise \end{cases}$$

<u>אופרטורים עבור תדלוק:</u>

בנוסף לנסיעה לצורך הורדת הזמנה, ניתן לבצע נסיעה לצורך תדלוק. במקרה זה שלומי נסע לצומת v_2 הנמצאת ברשימת תחנות הדלק $GasStations$. גם כאן נדרוש שהיה לנו מספיק דלק כדי להגיע לתחנה, אך מכאן והלאה מיכל הדלק שלו יהיה מלא.

לכל $j \in [l]$ נגדיר את האופרטור o_{f_j} באופן הבא:

$$\forall_{(u,d,T,F) \in S}: o_{f_j}((u, d, T, F)) \triangleq \begin{cases} (f_j, d_{refuel}, T, F) & ; \quad d \geq Dist(u, f_j) \\ \emptyset & ; \quad otherwise \end{cases}$$

לבסוף, קבוצת כל האופרטורים הינה:

$$O_d \triangleq \{o_{t_i} | i \in [k]\} \cup \{o_{f_j} | j \in [l]\}$$

- **עלות אופרטור:**

המחיר של מעבר בין מצב אחד למשנהו הוא המרחק בין הצמתים המיוצגים ע"י המצבים.

$$cost\left(((v_1, d_1, T_1, F_1), (v_2, d_2, T_2, F_2))\right) = Dist(v_1, v_2)$$

- **המצב ההתחלתי:**

$$I_d \triangleq (v_0, d_0, Ord, \emptyset)$$

- **מצבי המטרה:**

$$G_d \triangleq \{(u, d, \emptyset, Ord) \in S | u \in Ord\}$$

נגדיר עתה שני מדדים עבור מרחק בין זוג צמתים במפה $v_1, v_2 \in V$:

- $AirDist(v_1, v_2)$ – המרחק האווירי בין v_1, v_2.
- $MapDist(v_1, v_2)$ – אורך המסלול הקצר ביותר מ- v_1 ועד v_2 במפה.

כאמור, בעזרת הגדרת מרחב החיפוש ובשילוב עם שני מדדי מרחק שזה עתה הוגדרו, ניתן לקבל ייצוג לשתי בעיות שונות. הבעיה הראשונה, המושרת ע"י $MapDist$, מתארת את הבעיה הסופית שנרצה לפתור בתרגיל זה, כלומר - מציאת המסלול הקצר ביותר ברשת הכבישים. לבעיה זו נקרא $StrictDeliveries$.
בעיה נוספת, המושרת ע"י $AirDist$, אינה מתייחסת למפת הכבישים ולכן נסמנה ע"י $RelaxedDeliveries$.

תרגילים

2. מהם ערכי הקיצון (המקסימלי והמינימלי) האפשריים של מקדם הסיעוף במרחב החיפוש? נמקו בקצרה.
3. האם ייתכנו מעגלים במרחב המצבים שלנו? אם כן תנו דוגמה למעגל כזה, אחרת נמקו.
4. כמה מצבים יש במרחב זה? האם כולם ישיגים? נמקו.
5. האם ייתכנו בורות ישיגים מהמצב ההתחלתי שאינם מצבי מטרה במרחב המצבים?
6. הגדירו פורמלית את פונקציית העוקב $Succ: S \to P(S)$ (ללא שימוש בקבוצה O).
שימו לב, אנו מצפים לביטוי (או איחוד של ביטויים) מהצורה:
$$Succ\big((v_1, d_1, T_1, F_1)\big) = \left\{(v_2, d_2, T_2, F_2) \middle| \begin{matrix} d_2 = ?, T_2 = ?, F_2 = ? \\ v_2 \in ? \end{matrix}\right\}$$
7. בהנחה שאין שתי הזמנות במיקומים זהים, מהו חסם תחתון לעומק המינימאלי של מצב מטרה כלשהו במרחב החיפוש? נמקו.

חלק ד' – מתחילים לתכנת (7 נק')

הורידו את *ai_hw1.zip* מהאתר וטענו את התיקייה שבתוכו לסביבת העבודה המועדפת עליכם.

מבנה מפת הדרכים

את מפת הדרכים טוענים בעזרת קריאה לפונקציה *framework.ways.load_map_from_csv()* עם פרמטר שמציין את הנתיב של קובץ המפה. פונקציית הטעינה מחזירה אובייקט מטיפוס ***Roads***. בקובץ main.py שקיבלתם כבר כתובה קריאה לפונק' זו (עם שם קובץ המפות לפי נתיב יחסי בפרוייקט). הקריאה אמורה להמשיך לעבוד בצורה תקינה גם בלי פעולה נוספת מצדכם.

הטיפוס ***Roads*** יורש מ- *dict*; כלומר ***Roads*** הינו בבסיסו מיפוי ממזהה ייחודי של צומת במפה (מספר שלם) לאובייקט מטיפוס ***Junction*** שמייצג את אותו הצומת.

כל צומת הוא כאמור מטיפוס ***Junction***. לצומת יש את השדות הבאים: (1) מספר *index* ייחודי; (2+3) קואורדינטות lat, lon (קווי אורך ורוחב) של המיקום הגיאוגרפי של הצומת במפה; ו- (4) רשימה *links* המכילה את כל הקשתות לשכניו. כל קשת כזו מייצגת כביש במפה. קשת היא אובייקט מטיפוס ***Link*** עם מאפיינים source ו- target (המזהים של צמתי המקור והיעד של הקשת).

אינכם אמורים לשנות את הקוד של המפה, אלא רק להשתמש בו.

הכרת התשתית הכללית לייצוג ופתרון בעיות גרפים

המחלקות **GraphProblemState**, **GraphProblem** (בקובץ graph_search/graph_problem_interface.py) מגדירות את הממשק (interface) בו נשתמש על מנת לייצג מרחב מצבים. אלו הן מחלקות אבסטרקטיות – כלומר מוגדרות בהן מתודות שאינן ממומשות. לכן, בפרט, לא ניתן ליצור אובייקט מטיפוסים אלו. כדי להגדיר מרחב מצבים חדש יש לרשת משתי המחלקות הנ"ל. בהמשך התרגיל תראו דוגמא למימוש של מרחב מצבים באופן הנ"ל (שסיפקנו עבורכם) ותממשו 2 נוספים בעצמכם.

המחלקה **GraphProblemSolver** (באותו הקובץ) מגדירה את הממשק בו נשתמש בכדי לחפש בגרפים. למחלקה יש מתודה אבסטרקטית אחת בשם solve_problem() שמקבלת כפרמטר בעיה (אובייקט מטיפוס שיורש מ- GraphProblem) ומחזירה את תוצאות החיפוש (אובייקט מטיפוס SearchResult). כל אלג' חיפוש שנממש ישתמש בממשק הנ"ל (ירש ממחלקה זו או ממחלקה שיורשת ממנה).

שימו לב: אלגוריתמי החיפוש אותם נממש לאורך התרגיל יהיו כללים בכך שלא יניחו כלום על הבעיות אותן יפתרו, פרט לכך שהן תואמות לממשק המוגדר ע"י GraphProblemState, GraphProblem.

המחלקה **BestFirstSearch** (בקובץ graph_search/best_first_search.py) יורשת מהמחלקה GraphProblemSolver (שתוארה לעיל) ומייצגת אלגוריתמי חיפוש מהמחלקה Best First Search. כפי שנלמד בכיתה, אלו הם אלגוריתמים שמתחזקים תור עדיפויות בשם **open** של צמתים (פתוחים) הממתינים לפיתוח. כל עוד תור זה אינו ריק, האלג' בוחר את הצומת הבא בתור העדיפויות ומפתח אותו. המחלקה מממשת את המתודה _solve_problem() בהתאם. דוגמאות לאלגוריתמים ממשפחה זו: Uniform Cost, Greedy Best Search, A*. כאמור, Best First Search הינה משפחה של אלגוריתמי חיפוש (מכונה גם "אלגוריתם גנרי"), כלומר היא מגדירה שלד כללי של מבנה האלגוריתם, ומשאירה מספר פרטי מימוש חסרים. לכן, המחלקה BestFirstSearch אף היא אבסטרקטית. גם בה מוגדרות מספר מתודות אבסטרקטיות שעל היורש (אלגוריתם החיפוש הקונקרטי) לממש. המתודה האבסטרקטית _calc_node_expanding_priority() מאפשרת ליורש להגדיר את אופן חישוב ערך ה- f-score של צומת. כזכור, ערך זה משמש כעדיפות של צומת בתור העדיפויות open (אנו מכנים ערך זה בשם expanding priority). המתודה האבסטרקטית _open_successor_node() מאפשרת ליורש להגדיר את אופן הטיפול בצומת חדש שזה עתה נוצר ומייצג מצב עוקב של המצב המיוצג ע"י הצומת שנבחר אחרון לפיתוח (הכנסה ל- open, בדיקה ב- close). בנוסף, האלגוריתם מאפשר מצב של חיפוש-גרף כפי שנלמד בכיתה, ע"י תחזוק אוסף **סגור / close** של צמתים שכבר פיתחנו במהלך החיפוש (ה- constructor של BestFirstSearch מקבל פרמטר בולאני בשם use_close שקובע האם להשתמש ב- close).

מבנה הקלטים לבעיות המשלוחים ואופן טעינתם

המחלקה **DeliveriesProblemInput** (בקובץ deliveries/deliveries_problem_input.py) מייצגת קלט לבעיית המשלוחים. מחלקה זו אחראית לטעינה של קלטים שסיפקנו לכם כקבצי טקסט. הנה דוגמא לאופן השימוש במחלקה: big_delivery = DeliveriesProblemInput.load_from_file('big_delivery.in', roads). המחלקות שמייצגות את בעיות המשלוחים (נראה בהמשך) תקבלנה אובייקט מסוג זה. בקובץ הראשי main.py כבר כתובה שורת טעינה כזו במקומות הנדרשים. הבהרה: הקוד שקורא את קבצי הקלט כבר מוכן ואין צורך לממש קריאות נוספות מקבצים.

תרגילים

8. סעיף זה נועד על מנת להתחיל להכיר את מבנה הקוד.
   a. חלצו את תוכן התיקייה ai_hw1.zip.
   b. אם אתם משתמשים ב- IDE לכתיבת והרצת קוד פייתון (אנחנו ממליצים PyCharm), פתחו פרויקט חדש שתיקיית האם שלו היא התיקייה הראשית של קובץ ה- zip שחולץ (אמור להיות שם קובץ בשם main.py).
   c. פתחו את הקובץ main.py, קראו את החלק בקוד שמעליו מופיעה הערה המתאימה למספר סעיף זה. שורות קוד אלו מבצעות: יצירת בעיית מפה חדשה, יצירת אובייקט מסוג אלג' חיפוש uniform cost, הרצת אלג' החיפוש על הבעיה ולבסוף הדפסת התשובה שהתקבלה מההרצה. הריצו את הקובץ. וודאו שמודפסת לכם שורה למסך שמתארת את פתרון בעיית החיפוש במפה. זאת גם הזדמנות טובה לוודא שהחבילות numpy, matplotlib מותקנות אצלכם כראוי (אם התקנתם Anaconda בהתאם להסבר שסיפקנו במחברת colab מהתרגול הראשון אז הכל אמור לעבוד).
   d. פתחו את הקובץ deliveries/map_problem.py. בתוכו יש לכם שתי משימות (המסומנות ע"י הערות **TODO** כמו בעוד מקומות רבים לאורך המטלה). בשתי משימות אלו אתם מתבקשים לבצע שינוי בקוד של המחלקה MapProblem כדי לתקן ולהשלים את המימוש שסיפקנו לכם. קראו והבינו את מימוש המחלקה. שימו לב שמחלקה זו יורשת מהמחלקה GraphProblem (שתוארה מקודם) ומממשת את המתודות האבסטרקטיות הנדרשות.
   e. עתה, לאחר תיקון קוד המחלקה MapProblem, הריצו בשנית את main.py. הוסיפו לדו"ח את פלט הריצה המתוקנת.

חלק ה' – אלגוריתם A* (10 נק')

עתה נתחיל במימוש Weighted A*.

עיינו בקובץ graph_search/astar.py. שם מופיע מימוש חלקי למחלקה AStar. שימו לב: המחלקה AStar יורשת מהמחלקה האבסטרקטית BestFirstSearch (הסברנו עליה בסעיף ד'). זהו את החלק בהצהרת המחלקה AStar בו הירושה מוגדרת. המחלקה AStar צריכה לממש את המתודות האבסטרקטיות שמוגדרות ע"י

BestFirstSearch. הכותרות של מתודות אלו מופיעות כבר במימוש החלקי של המחלקה AStar, אך ללא מימושן. בסעיף זה נרצה להשלים את המימוש את המחלקה AStar ולבחון אותה.

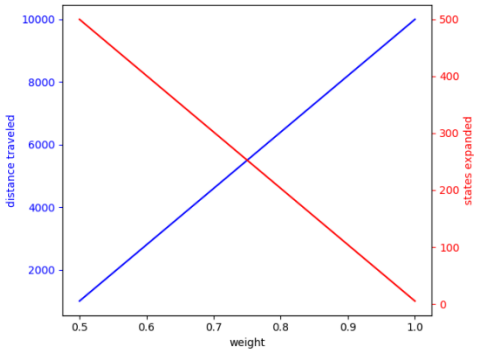
שימו לב: לאורך התרגיל כולו אין לשנות את החתימות של המתודות שסיפקנו לכם. בנוסף, אין לשנות קבצים שלא התבקשתם באופן מפורש.

תרגילים

9. השלימו את המשימות הדרושות תחת הערות ה- **TODO** בקובץ graph_search/astar.py כך שנקבל מימוש תקין לאלגוריתם Weighted A*, כפי שראיתם בהרצאות. בכדי להבין את מטרת המתודות השונות שעליכם לממש, הביטו במימוש המחלקה BestFirstSearch שעושה בהן שימוש. בנוסף, היעזרו במימוש שסיפקנו לכם ל- UniformCost (בקובץ graph_search/uniform_cost.py).
10. בכדי לבחון את האלג' שמימשתם זה עתה, השלימו את המשימות הדרושות תחת הערות ה- **TODO** הרלוונטיות לסעיף זה בקובץ main.py. כידוע, לצורך הרצת A* יש צורך בהיוריסטיקה. ה- constructor של המחלקה AStar מקבל את טיפוס ההיוריסטיקה שמעוניינים להשתמש בה. לצורך בדיקת שפיות, הפעילו את ה- A* על בעיית המפה שפתרתם בסעיף הקודם עם NullHeuristic (מסופקת בקובץ graph_search/graph_problem_interface.py. מחלקה זו אמורה להיות כבר מוכרת מ- main.py ללא צורך בביצוע import נוסף). וודאו שהתוצאה המודפסת זהה לזו שקבלתם בעזרת Uniform Cost.
11. כפי שראינו בהרצאות ובתרגולים, היוריסטיקה פשוטה לבעיית המפה היא מרחק אווירי לפתרון. היכנסו לקובץ deliveries/map_heuristics.py וממשו את ההיוריסטיקה הזו במחלקה AirDistHeuristic. כעת הריצו שוב את הבעיה שפתרתם בסעיף הקודם, אך כעת בעזרת ההיוריסטיקה (מלאו ב- main.py את המשימות שקשורות לסעיף זה). העתיקו לדו"ח את פלט הריצה.

שימו לב: בכדי לחשב מרחק בין זוג Junctions, **אין לחשב את המרחק האווירי ישירות על ידי קווי רוחב ואורך**, אלא יש להשתמש במתודה calc_air_distance_from() של המחלקה Junction.

12. כעת נרצה לבחון את השפעת המשקל w על ריצת A*. מלאו בקובץ main.py את המשימות הרלוונטיות לסעיף זה. ממשו את הפונק' run_astar_for_weights_in_range() שחתימתה מופיעה בקובץ main.py. פונק' זו מקבלת היוריסטיקה ובעיה לפתרון ומשתמשת באלג' A* בכדי לפתור את בעיה זו תוך שימוש בהיוריסטיקה הנתונה ועם 20 משקולות שונות בתחום הסגור [0.5,1]. את התוצאות של ריצות אלו היא אמורה לשמור ברשימות ולאחר מכן היא אמורה לקרוא לפונק' בשם plot_distance_and_expanded_wrt_weight_figure() (שגם אותה עליכם לממש). פונק' זו אחראית ליצור גרף שבו מופיעות 2 עקומות: אחת מהעקומות (הכחולה) מתארת את טיב הפתרונות (בציר y) כפונק' של המשקל (אורך המסלול). העקומה השנייה (האדומה) מתארת את מספר המצבים שפותחו כפונק' של המשקל. השתמשו ב- run_astar_for_weights_in_range() מהמקום הרלוונטי ב- main.py (מספר סעיף זה מצוין במקום זה) ע"מ ליצור את הגרף המתאים עבור פתרון בעיית המפה תוך שימוש בהיוריסטיקה AirDistHeuristic. צרפו את הגרף שנוצר לדו"ח. הסבירו את הגרף שהתקבל. על הגרף להראות כמו בדוגמה הזו (צורת העקומות עצמן עשויה להשתנות כמובן):



חלק ו' – בעיית המשלוחים המופשטת (10 נק')

כעת נרצה לממש את בעיית המשלוחים המופשטת. בבעיה זו נרצה למצוא סדר אופטימאלי לפיזור הזמנות תוך התחשבות באילוצי הדלק אך ללא התייחסות למערכת הכבישים במפה. בפועל, אנו נניח שהעלות מצומת לצומת הינה המרחק האווירי המקשר בין הצמתים.

13. התבוננו בקובץ deliveries/relaxed_deliveries_problem.py וממשו את RelaxedDeliveriesState ואת RelaxedDeliveriesProblem.

לצורך הרצת A* על בעיה זו, יש צורך להגדיר היוריסטיקה. נבחר את ההיוריסטיקה הבאה:

$$h(s) = \max_{i \in T} AirDist(s.v, t_i)$$

ונסמנה ב $MaxAirDist$.

14. רשמו בדו"ח האם היוריסטיקה זו קבילה? נמקו.
15. כעת גשו לקובץ deliveries/deliveries_heuristics.py וממשו את היוריסטיקה $MaxAirDistHeuristic$.
16. בקובץ main.py, תחת ההערה שמתאימה לסעיף זה, מלאו והריצו את קטע הקוד שמפעיל את אלג' ה- A* על בעיה זו עם ההיוריסטיקה שתוארה על הקלט big_delivery. הוסיפו לדו"ח את פלט הריצה.

היוריסטיקה מתקדמת

נשתמש בהיוריסטיקה קבילה המבוססת על עבודה של Christofides מ-1976 מתחום אלגוריתמי הקירוב.

הרעיון הכללי: נשים לב שבמצב שאינו מצב מטרה, יש לנו רשימה של צמתי דרכים שעלינו לבקר עדיין. אם נקל את הבעיה ונניח שאנו יכולים לעבור צמתים אלה ישירות בעלות של המרחק האווירי ביניהם, נקבל קליקה לא מכוונת עם משקלים על הקשתות. ועכשיו עלינו למצוא את המרחק המינימלי שנדרש כדי לבקר בכל הצמתים האלה (ללא הגבלה על מספר הביקורים).

איך מוצאים את המרחק המינימלי לביקור בכל הצמתים של גרף לא מכוון? צודקים! עם עץ פורש מינימום!

ניתן להסיק שמשקל העפ"מ של הקליקה שתוארה הוא חסם תחתון להיוריסטיקה מושלמת מכל מצב, אך לצערנו בגרף מכוון עם אילוצים על סדר הביקור כמו שלנו, היוריסטיקה הזו עלולה להיות לא מאוד מיודעת. תכף ניווכח שלמרות זאת, ההיוריסטיקה תחסוך לנו משמעותית במספר הפיתוחים.

תרגיל

17. בקובץ main.py, תחת ההערה שמתאימה לסעיף זה, השלימו והריצו את קטע הקוד שמפעיל את אלג' ה- A* על הבעיה relaxed deliveries עם הקלט big_delivery ועם ההיוריסטיקה $MSTAirDistHeuristic$ (ההיוריסטיקה כבר ממומשת במלואה עבורכם בקובץ deliveries/deliveries_heuristics.py ואמורה להיות מוכרת ב- main.py ללא צורך בביצוע import נוסף). הוסיפו לדו"ח את פלט הריצה.
18. בקובץ main.py, תחת ההערה שמתאימה לסעיף זה, השלימו והריצו את קטע הקוד המשווה את ריצת האלגוריתם A* עם משקולות משתנים (כפי שתואר בסעיף 12). פתרו את הבעיה relaxed deliveries עם הקלט big_delivery והשתמשו בהיוריסטיקה MSTAirDistHeuristic. הוסיפו לדו"ח את הגרף שנוצר.

שימו לב: הסעיפים האחרונים יכולים לעזור לכם לוודא שהאלגוריתמים שלכם אכן עובדים כשורה. ודאו שהתוצאות שקיבלתם הגיוניות.

לצורך הניסויים הבאים נמשיך להשתמש בהיוריסטיקה MSTAirDistHeuristic.

חלק ז' – אלגוריתם חיפוש חמדני-סטוכאסטי (20 נק')

כעת נבחן הרצה של אלגוריתם Anytime באמצעות אלגוריתם חמדני-סטוכסטי ונשווה את ביצועיו לאלגוריתם A* ולאלגוריתם Greedy Best First.

ראשית, נזכר כי באלגוריתם חמדני-דטרמיניסטי (Greedy Best), אנו בוחרים את הצומת הבא לפיתוח רק על פי הערך ההיוריסטי שלו. למזלנו, מאחר ומימשנו Weighted A*, יש לנו ביד מימוש "בחינם" גם ל- greedy best ע"י שימוש במשקל $w = 1$.

בדומה לאלגוריתם החמדני-דטרמיניסטי, גם באלגוריתם הסטוכאסטי נשתמש בהיוריסטיקה בלבד לצורך בחירת הצומת הבא לפיתוח. עם זאת, במקום לבחור באופן דטרמיניסטי את המצב הבא לפיתוח, נבחר אותו באקראי מתוך N מצבים הטובים ביותר (בעלי ההיוריסטיקה הנמוכה ביותר) באותו הרגע.

באלגוריתם אותו נממש בחלק זה, נשתמש בערך הקבוע $N = 5$.

לצורך הבחירה האקראית, יש להגדיר פונקציית הסתברות על סט מצבים כלשהו.

לשם נוחות הכתיבה, נגדיר וקטור עזר $x \in [0,1]^{min(N,|OPEN|)}$ המכיל ניקוד חיובי (גם כאן נשתמש במרחק האווירי) של N המצבים בעלי הערך ההיוריסטי הקטן ביותר ב OPEN:

$$x^t = [h_1, h_2, \dots, h_{min(N,|OPEN|)}]$$

$$\forall x_i \in x^t: \Pr(x_i) = \frac{(x_i)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} (x_h)^{-\frac{1}{T}}}$$

לשם יציבות נומרית נשנה את הסקאלה של הניקוד ע"י הגדרת $\alpha = \min_j x_j$ ועדכון ההסתברויות:

$$\forall x_i \in x^t: \Pr(x_i) = \frac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} \left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}}$$

קל לראות שמתקבלת התפלגות חוקית (כל הסתברות היא בין 0 ל-1, וסכום כל ההסתברויות הוא 1).

למשל, עבור הוקטור x^T הנ"ל והקבועים $N = 5, T = 0.5$ נקבל וקטור ההסתברויות P:

$$x^t = [400, 450, 900, 390, 550]\ ,\ P^t \cong [0.28, 0.22, 0.05, 0.3, 0.15]$$

הגישה הסטוכאסטית

בגישה זו נרצה להריץ את האלגוריתם המון פעמים, ולהחזיר את הפלט הטוב ביותר שקיבלנו במהלך הריצה. יתכן שמספר הפעמים יהיה קבוע מראש (כמו בתרגיל בהמשך), ויתכן שנריץ את האלגוריתם ככל שיאפשרו לנו, עד שגורם חיצוני (למשל המשתמש) ידרוש מאתנו לעצור ולהחזיר את התוצאה הטובה ביותר שקיבלנו עד כה (אלגוריתם anytime).

הטמפרטורה

ניתן להתייחס למשתנה T כמו "טמפרטורה" במערכת. ככל שהוא גדל, אנחנו מכניסים יותר "רעש" ומאפשרים בחירות יותר "הרפתקניות", וככל שהוא קטן נעדיף בחירות יותר "בטוחות". נרחיב על גישה זו בהמשך הקורס.

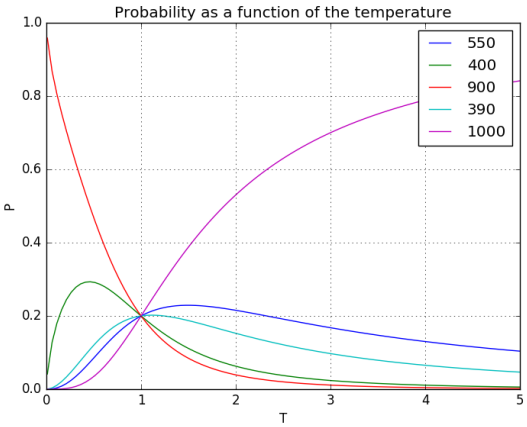
הערה: בפונקציית ההסתברות שלעיל המשתנה T לא מתאים במדויק לפרשנות הפיזיקלית של טמפרטורה. יש פונקציות אחרות בהן זה כן מתקיים, כמו למשל פונקציית בולצמן.

באלגוריתם אותו נממש בחלק זה, נתחיל בערך $T = 1$ ונקטין אותו פי 0.95 בכל שלב (משמע בכל פעם בה נחשב את N האיברים הכי טובים).

תרגילים

19. הוכיחו ששינוי הסקאלה לא משנה את ההתפלגות לוקטור x נתון.
20. השלימו את הקוד בקובץ *experiments/temperature.py*, כך שיתקבל הגרף הבא:

לכל אחד מהאיברים בוקטור x^t הנתון לעיל במסמך זה, ציירו בעקומה נפרדת את פונקציית ההסתברות שלו כפונקצייה של המשתנה T, עבור 100 ערכי T מ- 0.01 עד 5 (במרווחים שווים).
יש להוסיף מקרא לגרף.
על הפלט להיראות כמו בדוגמה הנלווית (העקומות עצמן הן להמחשה בלבד).



צרפו את הגרף לדו"ח.

21. על סמך התבוננות בגרף, או ניתוח אנליטי של פונקציית ההסתברות שהוגדרה, הסבירו איך יתנהג האלגוריתם בגבול $T \to 0$.
22. על סמך התבוננות בגרף, או ניתוח אנליטי של פונקציית ההסתברות שהוגדרה, הסבירו איך יתנהג האלגוריתם בגבול $T \to \infty$.
23. השלימו את החלקים החסרים במחלקה GreedyStochastic בקובץ graph_search/greedy_stochastic.py.
24. בקובץ main.py, תחת ההערה שמתאימה לסעיף זה, השלימו והריצו את קטע הקוד שמבצע 100 הרצות של האלגוריתם הסטוכאסטי הגרידי על הבעיה relaxed deliveries עם הקלט big_deliveries. בסעיף זה השתמשו בהיוריסטיקה MSTAirDistHeuristic. שמרו ברשימה את העלויות של התוצאות שנמצאו בריצות אלו.
כתבו קוד המציג גרף של טיב הפתרון של האלג' כפונקציה של מספר האיטרציה. המדד לטיב הפתרון בו נשתמש הוא המרחק הכולל במסלול המוחזר ע"י האלגוריתם (עלות התוצאה).
האלגוריתם Anytime Greedy Stochastic מריץ את האלגוריתם גרידי-סטוכסטי k פעמים (אצלנו k=100), ולאחר כל הרצה שומר את הפתרון הטוב ביותר שנמצא עד כה. הציגו באותו הגרף גם את תוצאת האלג' ה- anytime (העקומה המוצגת עבורו אמורה להיות <u>מונוטונית יורדת</u>).
בנוסף, הוסיפו לאותו הגרף עקומה (קבועה) של טיב הפתרון של האלגוריתם החמדני-דטרמיניסטי וכל עקומה עבור A* (המתקבלות על ידי שתי הרצות A* עם w=0.5 וכן w=1). צרפו את הגרף לדו"ח.

*** בשלב כתיבת הקוד והדיבאג מומלץ לעבוד עם קבוע קטן מ-100 כדי לחסוך זמן ***

חלק ח' – אלגוריתם מבוסס A* (20 נק')

בחלק זה נריץ אלגוריתם A* כדי לפתור את הבעיה עצמה שלשמה התכנסנו (StrictDeliveries).

מתוך הגדרת מרחב המצבים והמימוש הכללי של A*, אין מניעה שנשתמש באלגוריתם זה כדי למצוא פתרון אופטימלי במרחב המצבים של השליח. כדי להבטיח שנקבל פתרון אופטימלי, נפעיל את A* על מרחב המצבים של השליח, ולחישוב עלות היורשים של כל מצב נשתמש ב- A* על מפת הדרכים.

נשים לב ש A* עם היוריסטיקה אופטימית (קבילה) הוא אלגוריתם קביל, ולכן אמור למצוא את הפתרון **<u>האופטימלי</u>**. מכיוון שהבעיה היא NP קשה, אנחנו לא יכולים לצפות שהאלג' יעבוד באופן כללי בזמן סביר, אפילו עם היוריסטיקות קבילות מאוד מתוחכמות (אחרת נוכיח ש- P=NP, נקבל פרס טיורינג ונלך הביתה). לכן נצמצם את השיח בחלק זה אך ורק ל- dataset קטן.

שימו לב: לאורך כל הפרק, עבור ה- A* הפנימי שרץ על בעיית המפה, השתמשו בהיוריסטיקת AirDistHeuristic.

Caching

כפי שתיארנו, בכל פיתוח בריצת ה A* הראשית על בעיית המשלוחים, נצטרך לפתור גם ריצת A* בבעיית המפה. על מנת לחסוך חישובים חוזרים של אותם המסלולים על המפה, נרצה להוסיף מנגנון מטמון למחלקה שלנו שתזכור את המסלולים האופטימאליים בין צמתים שונים במפה.

המתודות *_get_from_cache, _store_in_cache* כבר מוכנות עבורכם במחלקת *StrictDeliveriesProblem*.

כל שעליכם לעשות הוא להוסיף קריאה למתודת השמירה *_store_in_cache(..)* עם התוצאה במקום המתאים בקוד וקריאת ערך מה - cache ע"י המתודה _get_from_cache(...).

שימו לב: על מנת להפעיל את מנגנון המטמון יש ליצור אובייקט *StrictDeliveriesProblem* עם ארגומנט *use_cache=True*.

תרגילים

25. היכנסו לקובץ deliveries/strict_deliveries_problem.py והשלימו את המימוש הנדרש במחלקות StrictDeliveriesState ו- StrictDeliveriesProblem לתיאור בעיית המשלוחים.
26. בקובץ main.py, תחת ההערה שמתאימה לסעיף זה, השלימו והריצו את קטע הקוד המשווה את ריצת האלגוריתם A* עם משקולות משתנים (כפי שתואר בסעיף 12). פתרו את הבעיה strict deliveries עם הקלט small_delivery והשתמשו בהיוריסטיקה MSTAirDistHeuristic. הוסיפו לדו"ח את הגרף שנוצר.
27. הציעו היוריסטיקה לפתרון בעיית ה StrictDeliveries המשתמשת בבעיית ה RelaxedDeliveries. הסבירו בדו"ח מדוע היוריסטיקה זו מיודעת. ממשו היוריסטיקה מתאימה במחלקה RelaxedDeliveriesHeuristic בקובץ deliveries/deliveries_heuristics.py.

28. בקובץ main.py, תחת ההערה שמתאימה לסעיף זה, השלימו והריצו את קטע הקוד שמשתמש בהיוריסטיקה זו כדי לפתור את הבעיה strict deliveries על הקלט small_delivery. השוו את מספר המצבים שפותחו לעומת הריצה בסעיף הקודם. האם זמן הריצה השתפר? הסבירו.

חלק ט' – הגשת המטלה (5 נק')

מעבר למימוש ולדו"ח, ציונכם מורכב גם מהגשה תקינה של המטלה לפי הכללים הבאים:

- **יש לכתוב קוד ברור**:
  - קטעי קוד מסובכים או לא קריאים יש לתעד עם הערות.
  - לתת שמות משמעותיים למשתנים.
- **הדו"ח**:
  - יש לכתוב בדו"ח את תעודות הזהות של **שני** המגישים.
  - הדו"ח צריך להיות מוקלד במחשב ולא בכתב יד.
  - יש לשמור על סדר וקריאות גם בתוך הדו"ח.
  - יש לתת כותרות מתאימות לגרפים ולצירים.
  - אלא אם נכתב אחרת, תשובות ללא נימוק לא יתקבלו.
  - יש לענות על השאלות לפי הסדר והמספרים שלהם.
- **ההגשה**:
- יש להעלות לאתר קובץ zip בשם AI1_123456789_987654321.zip (עם תעודות הזהות שלכם במקום המספרים).
- בתוך ה- zip צריכים להיות זה לצד זה:
  - הדו"ח הסופי בפורמט pdf בשם: AI1_123456789_987654321.pdf.
  - תיקיית הקוד AI1 שקיבלתם בתחילת המטלה, עם כל השינויים הנדרשים.

  נא לא להכניס לזיפ את התיקייה db שבתיקייה שקיבלתם.

חריגה מהכללים האלו עלולה לגרור הורדה של כל 5 הנקודות.

קוד לא ברור / לא עובד אף עלול להביא להורדה של נקודות נוספות בפרק בו הוא נכתב.

שימו לב: הקוד שלכם יבדק ע"י מערכת בדיקות אוטומטיות. במידה וחלק מהבדיקות יכשלו, הניקוד עבורן יורד באופן אוטומטי.

שימו לב: **העתקות תטופלנה בחומרה.** אנא הימנעו מאי-נעימויות.

פרק שני – שאלה תאורטית (10 נק')

יש לענות על פרק זה בסוף הדו"ח המוגש.

הערה: בתרגיל זה הניחו שהיוריסטיקה 1 מיודעת מהיוריסטיקה 2 כאשר לכל צומת מתקיים אי שיוויון חלש: $h_1(s) \geq h_2(s)$.

נתון מרחב מצבים ונתונה פונקציה היוריסטית חלקית h.

היוריסטיקה h מעל אוסף מצבים S תיקרא חלקית אם קיימת קבוצה $S' \subset S$ כך ש-h אינה מוגדרת על S'. ברשותכם פרדיקט $Applicable_h: S \to \{True, False\}$ המחזיר $True$ אמ"מ h מוגדרת על s.

נניח שפונקציית המחיר חיובית וחסומה ע"י $\delta > 0$ וש-h קבילה ואי-שלילית במצבים עליהם היא מוגדרת.

רוצים להפעיל A* על מרחב המצבים עם היוריסטיקה h, אך לא ניתן לעשות זאת מכיוון שהיא לא מוגדרת על כל המצבים. פתרון בסיסי לבעיה הוא שימוש בהיוריסטיקה החילופית הבאה:

$$h_0(h,s) = \begin{cases} h(s), & Applicable_h(s)\ is\ True \\ 0, & otherwise \end{cases}$$

א. הוכיחו: אם h קבילה גם h_0 קבילה.

ב. נתון שמרחב המצבים הינו עץ. מצאו פתרון מיודע יותר מהפתרון הבסיסי הנתון אשר שומר על קבילות. כתבו את כל הפסאודו-קוד של האלגוריתם.
הבהרה:
- בהקשר הנוכחי, נאמר שפתרון מיודע יותר מפתרון אחר אם הוא מסתמך על היוריסטיקה מיודעת יותר.

ג. כעת לא נתון דבר על טופולוגיית מרחב המצבים. חזרו על סעיף ב'.

ד. כעת נתונה היוריסטיקה חלקית h' הנותנת בדרך קסם את הערך המושלם עבור מצב ההתחלה בלבד, ואינה מוגדרת על יתר המצבים. הניחו שמרחב המצבים הוא עץ מכוון.
הוכיחו/הפריכו: קיים אלגוריתם קביל המשתמש ב-h' וזמן ריצתו חסום מלעיל ע"י A^* המשתמש בהיוריסטיקה $h_0(h',s)$ (הכוונה שאלגוריתם כזה מהווה שיפור ל-A* עם $h_0(h',s)$).

שימו לב: היוריסטיקה h מתחילת השאלה לא נתונה בסעיף זה.